(2)

Performance of Redundant Disk Array Organizations in Transaction Processing Environments*†

Antoine N. Mourad‡ W. Kent Fuchs†† Daniel G. Saab††

‡AT&T Bell Laboratories
101 Crawfords Corner Rd
Holmdel, NJ 07733
mourad@hoserve.att.com

††Coordinated Science Laboratory
University of Illinois
Urbana, Illinois 61801
{fuchs,saab}@crhc.uiuc.edu

Contact: Antoine Mourad
Ph. (908) 949-1694
Fax (908) 834-5906

September 16, 1993

Abstract

Disk arrays provide high data transfer rates by striping data over multiple disks. They also balance the load among disks in the same array. Redundant arrays use parity information to allow recovery from media failures in systems requiring high availability. In transaction processing environments, the high transfer rates of disk arrays are not fully exploited because I/O requests are typically small. However, redundant arrays are especially useful in such environments because they achieve media recovery at a significantly lower storage cost than mirrored disks.

The performance of two redundant array organizations, data striping with rotated parity (RAID5) and parity striping, is analyzed and compared to that of mirrored disk systems and systems using no striping and no redundancy. Trace data from large scale commercial transaction processing systems are used to evaluate the performance of the organizations. Data and parity caching mechanisms for reducing the write penalty in arrays using parity are investigated and their effect on performance is analyzed.

Keywords: Parallel I/O, Disk Arrays, Transaction Processing, Media Recovery.

*This research was supported in part by the National Aeronautics and Space Administration (NASA) under Contract NAG 1-613 and in part by the Department of the Navy and managed by the Office of the Chief of Naval Research under Grant N00014-91-J-1283.
†Portions of this paper appeared in the proceedings of the 1993 International Conference on Parallel Processing.

1 Introduction

Striped disk arrays have been proposed and implemented for increasing the transfer bandwidth in high performance I/O subsystems [1–4]. In order to allow the use of a large number of disks in such arrays without compromising the reliability of the I/O subsystem, redundancy is included in the form of parity information. Patterson et al. [5] have presented several possible organizations for Redundant Arrays of Inexpensive Disks (RAID).

Reliable storage is a necessary feature in transaction processing systems requiring high availability. Media failure in such systems is traditionally dealt with by periodically generating archive copies of the database and by logging updates to the database performed by committed transactions between archive copies into a redo log file. When a media failure occurs, the database is reconstructed from the last copy and the log file is used to apply all updates performed by transactions that committed after the last copy was generated. A media failure can cause significant down time and the overhead for recovery is quite high. For large systems, e.g., with over 150 disks, the mean time to failure (MTTF) of the permanent storage subsystem can be less than 28 days[1]. Mirrored disks have been employed to provide rapid media recovery [6]. However, disk mirroring incurs a 100% storage overhead which is prohibitive in many cases. Redundant disk array organizations [5, 7] provide an alternative for maintaining reliable storage[2].

In this paper, we use two traces from commercial transaction processing sites to evaluate the performance of two redundant disk array organizations (RAID5 and Parity Striping) and compare it to that of mirrored disks and non-striped, non-redundant systems. We evaluate the performance of non-cached organizations as well as organizations using a non-volatile cache in the controller. Caches provide two major benefits: one comes from taking advantage of the locality in I/O accesses to reduce the number of actual disk accesses and the other results from buffering write requests in the cache which reduces the effect of the write penalty on the response time seen by the user. We also examine the benefits of using parity caching as a way to further reduce the effect of the high cost of individual writes and we compare a RAID4 organization that caches both parity and data in the same non-volatile cache to a RAID5 organization that caches only data. Using I/O traces from commercial applications to drive our disk array simulations allows us to properly account for various workload characteristics such as: the I/O arrival process, the amount of skew in the access distribution among disks and within a given disk, and the amount of temporal and spatial locality.

[1] Assuming an MTTF of 100,000 hours for each disk.

[2] However, even when disk mirroring or redundant disk arrays are used, archiving and redo logging may still be necessary to protect the database against operator errors or system software design errors.

Chen et al. [8] compared the performance of RAID0,[3] RAID1,[4] and RAID5[5] systems. They used a synthetic trace made of a distribution of small requests representing transaction processing workloads combined with a distribution of large requests representing scientific workloads and disk measurements on an Amdahl 5890. Gray et al. [7] proposed the Parity Striping organization and used analytical models to derive the minimum (zero load) response time and the throughput at 50% utilization for fixed size requests. Their results suggest that Parity Striping is more appropriate than RAID5 for database and transaction processing systems. Their model does not take into account the effect of skew in the distribution of accesses to disks, which turns out to be an important element in favor of RAID5. Chen and Towsley [9] developed queuing models for comparing the performance of RAID5 and parity striping. Lee and Katz [10] compared the performance of various parity placements for RAID5 organizations. Menon and Mattson [11] analyzed the performance of RAID5 systems in the transaction processing environment using analytical models. They compared the performance of arrays made of different size building blocks and studied the effect of caching. Reddy [12] analyzed the effect of various parameters and policies in the design of a non-volatile I/O cache for systems where the cost of writes is higher than the cost of reads. He does not assume any particular array organization and the effect of the parity update traffic on read miss access time is not modeled. Ganger et al. [13] studied the benefits of data striping in reducing disk load imbalance and showed that it performs better than conventional data placement schemes.

In our evaluation of cached systems, we concentrate on comparing the behavior of the various array organizations when an I/O cache is used. Both read miss accesses and write (destage) accesses to data and parity are simulated. Bhide and Dias [14] have analyzed the RAID4 system with parity buffering in OLTP environments using analytical models. Their model suggests that a relatively large amount of non-volatile memory is necessary. We found that this is not the case in the I/O traces examined in this paper. They also propose an alternate scheme which writes parity updates sequentially on a log disk and then periodically writes them back to the parity disk. The log-based scheme uses up to four extra disks per array. The RAID 7 disk array system built and marketed by Storage Computer [15] uses the RAID4 disk organization with data and parity caching. Stodolsky et al. [16] proposed a parity logging scheme in which parity and log regions are distributed over the disks in the array.

The following section introduces the redundant disk array organizations discussed in this paper. Section 3 describes the trace data and the system model used in our simulations. In Section 4, we present the

[3]Data striping without redundancy.
[4]Data striping with mirroring.
[5]Data striping with rotated parity.

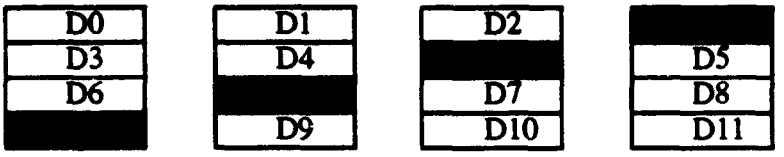


Figure 1: RAID5 with four disks.

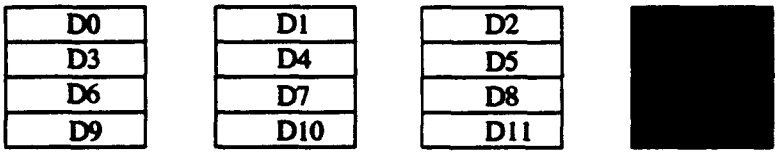


Figure 2: RAID4 with four disks.

experiments conducted and discuss the results.

2 Redundant Disk Arrays

2.1 Data Striping

One interesting organization for transaction processing environments is RAID with rotated parity (RAID5) in which blocks of data are interleaved across N disks while the parity of the N blocks is written on the $(N + 1)$st disk. The parity is rotated over the set of disks in order to avoid contention on the parity disk. Figure 1 shows the data and parity layout in a RAID5 organization with four disks. This pattern is repeated for the next set of blocks. An important parameter in the RAID5 organization is the striping unit. The striping unit can be defined as the "maximum amount of logically contiguous data stored on a single disk" [17].

The RAID5 organization allows both large (full stripe) concurrent accesses or small (individual disk) accesses. For a small write access, the data block is read from the relevant disk and modified. To compute the new parity, the old parity has to be read, XORed with the new data and XORed with the old data. The new data and new parity can then be written back to the corresponding disks.

The RAID4 organization shown in Figure 2 is similar to the RAID5 organization except for the fact that the parity for the N data disks is written on one parity disk. One disadvantage of the RAID4 organization is that the parity disk may become a bottleneck. The RAID4 organization becomes attractive when a non-volatile cache is used in which case parity blocks can be cached and written asynchronously to the parity disk.

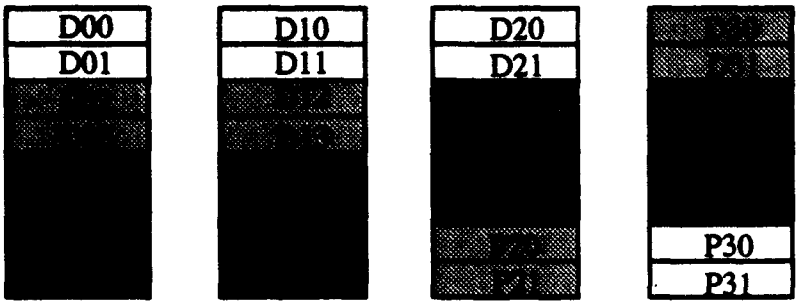


Figure 3: Parity striping of disk arrays.

2.2 Parity Striping

Gray et al. [7] studied ways of using an architecture such as RAID in transaction processing systems. They argued that because of the nature of I/O requests in OLTP systems, namely a large number of small accesses, it is not convenient to have several disks servicing the same request. In other words, since in transaction processing systems I/O time is dominated by seek time and rotational latencies rather than by transfer time, it is not advantageous to have a request spread over multiple disks because that will make all those disks spend a significant amount of time seeking and rotating in order to decrease an already small transfer time. Hence, the organization shown in Figure 3 was proposed. The shading in the figure indicates the areas that belong to the same parity group. It is referred to as parity striping. It consists of reserving an area for parity on each disk and writing data sequentially on each disk without interleaving. For a group of $N + 1$ disks, each disk is divided into $N + 1$ areas one of these areas on each disk is reserved for parity and the other areas contain data. N data areas from N different disks are grouped together in a *parity group* and their parity is written on the parity area of the $(N + 1)$st disk.

3 Workload and System Model

3.1 Trace Data

To evaluate the different redundant disk array organizations, we have used data from operational transaction processing systems, from IBM customer sites. We use one very large trace containing over 3.3 million I/O requests accessing an active database residing on 130 data disks and a second trace from a smaller system containing about 70 thousand I/O's and accessing an active database consisting of 10 data disks. The traces were collected using a low overhead tracing facility at installations running the DB2 database management system. The trace entries contain the absolute address of the block accessed, the type of access (read, write) and the time since the previous request. The time field is set to zero when both accesses are part of the same

Table 1: Disk and channel parameters.

Rotation speed	5400 rpm
Average seek	11.2 ms
Maximal seek	28 ms
Tracks per platter	1260
Sectors per track	48
Bytes per sector	512
Number of platters	15
Channel transfer rate	10 MB/s

Table 2: Trace characteristics.

	Trace 1	Trace 2
Duration	3hr 3min	1hr 40min
# of disks	130	10
# of I/O accesses	3,362,505	69,539
# of blocks transferred	4,467,719	143,105
# of single block reads	2,977,914	48,339
# of single block writes	312,961	17,557
# of multiblock reads	47,324	2,029
# of multiblock writes	24,306	2,098

multiblock request.

3.2 System Model

Using these data, we simulate the behavior of the I/O subsystem. We account for all channel and disk-related effects but we ignore cpu and controller processing times. The disk parameters used in the simulations are shown in Table 1. The total capacity of each disk is about 0.9 GByte. To compute the seek time as a function of the seek distance, we use a non-linear function of the form $a\sqrt{x-1} + b(x-1) + c$, x denoting the seek distance. Table 2 shows the characteristics of the traces used. We see that 98% of the accesses in Trace 1 and 95% of the accesses in Trace 2 are single-block accesses. The percentage of writes is 10% for Trace 1 and 28% for Trace 2. In addition to having a higher write fraction, Trace 2 exhibits more disk access skew than Trace 1. In terms of access locality, Trace 2 has less locality and larger working sets that Trace 1. This is partly due to the presence of a small amount of ad-hoc queries in the workload mix of Trace 2.

We compare four different organizations: *Base*, *Mirror*, *RAID5*, and *Parity Striping*. In the Base organization, disks are accessed independently without any striping or redundancy. The disks are divided

Table 3: Disk array organizations.

Non-cached organizations		Base
		Mirrored disks
		RAID5
		Parity Striping
Cached organizations	Data caching	Base
		Mirrored disks
		RAID5
		Parity Striping
	Data & parity caching	RAID4

into arrays of equal capacity. Each array can hold the equivalent capacity of N independent data disks. In the Base organization, there are N disks per array. In the Mirror organization, an array consists of $2N$ disks. In the case of the RAID5 and Parity Striping organizations, data and parity in each array are spread over $N + 1$ disks. Each array has one controller and an independent channel connecting it to the host. When comparing the various organizations, we make equal capacity comparisons as opposed to equal cost comparisons. We basically assume that we have a given database that has to be stored and that, for each organization, only the minimum number of disks needed to store the data is used. Therefore, the Mirror organization uses twice as many disks as the Base organization while a RAID5 or Parity Striping organization with $N + 1$ disks per array uses $N + 1/N$ times the number of disks in the Base organization. For RAID5 and Parity Striping, the total number of disks used changes with N. For Trace 1 and $N = 5$, RAID5 and Parity Striping use 26 arrays containing 6 disks per array or a total of 156 disks while, for $N = 10$, 13 arrays containing 11 disks per array or a total of 143 disks are used.

We compare the organizations both with cached controllers and non-cached controllers. In the case of cached controllers, we also consider a RAID4 organization that uses $N + 1$ disks per array: N disks for data and one for parity. Table 3 shows the various organizations considered in our study. No spindle synchronization is assumed. Block size is 4 kBytes. For the RAID5 and parity striping organizations, we study the effect of various parameters such as the striping unit in RAID5, the placement of the parity areas in Parity Striping, and the policy used to synchronize the parity access and the corresponding data access(es) within an update request.

3.3 Synchronization

For parity organizations (RAID5 and Parity Striping), when updating a single block or a portion of a stripe (less than half a stripe), the old data and old parity have to be read to compute the new parity. The access to the parity disk consists of reading the old parity block waiting for a full rotation and then writing the new parity block at the same location as the old. However, the write to the parity disk cannot occur until the old data have been read and the new parity has been computed. If one or more of the data disks has not completed the read operation for the old data by the time the head of the parity disk comes back to the parity block location, then the parity cannot be written and another full rotation time will be spent before the parity write can be performed. This can occur more than once if one of the data disks is very busy.

We compared five different strategies for handling the synchronization between the parity disk and the data disks. The first strategy is Simultaneous Issue (SI) in which the parity access is issued at the same time as the accesses to data and if the old data are not available by the time the parity disk reads the old parity and accomplishes a complete rotation, then the parity disk is held for the duration of some number of full rotations until the old data have been read. The second strategy is Read First (RF) and consists of waiting for the old data to be read before issuing the parity access. This strategy minimizes disk utilization but it might unnecessarily increase the response time of update requests. Another strategy that also minimizes disk utilization without unduly increasing update response time is the Read First with PRiority (RF/PR) method, which waits for the old data to be read before issuing the parity access, but it gives the parity access higher priority than non-parity accesses queued at the same disk. The fourth strategy consists of waiting for the data access(es) to reach the head of the queue and acquire the corresponding disk(s) before issuing the parity request. This strategy is called Disk First (DF). This strategy reduces the response time of the update access compared with the RF policy but may increase disk utilization slightly since the parity access could finish reading the old parity block and perform a full rotation before the read of the old data is completed. A variation on the DF policy consists of giving parity requests priority over other requests. This is called Disk First with PRiority (DF/PR). An analytical model for the performance of the DF/PR policy was developed by Chen and Towsley [9].

3.4 Buffering and Caching

In non-cached organizations, we assume that a number of track buffers is used in the controller to reduce the effects of channel contention on performance. Write data are transferred on the channel to the buffers and when the disk head arrives at the appropriate location they are written to the disk surface. Similarly

reads are transferred from the disk to the buffers and when the channel is available they are sent to the host. This avoids having to wait an extra rotation if the disk head is at the appropriate location but the channel is busy. The buffers are also used to hold the old data and parity that are read from disk in order to compute the new parity. The number of track buffers in the controller is proportional to the number of disks in the array attached to the controller. In our simulations we use five track buffers per disk.

In cached organizations, non-volatile memory should be used in the controller cache to protect against the loss of write data in the event of a power failure. If volatile memory is used, then the cache should be flushed frequently to reduce the extent of data loss when a failure occurs [18]. There is one cache per array. Read hits are satisfied from the cache. The response time for a read hit is equal to the response time (waiting time and transfer time) at the channel. On a read miss the block is fetched from disk. If the replaced block is dirty, it has to be written to disk. The cache replacement policy is LRU. On a write hit, the block is simply modified in the cache. In organizations using parity (RAID5 and Parity Striping), when a block is modified, the old data are kept in the cache to save the extra rotation needed to read the old data when writing the block back to disk. The old parity still has to be read and an extra rotation is required at the disk holding the parity. On a write miss, the block is written to the cache and the block at the head of the LRU chain is replaced. A background *destage* process groups consecutive blocks and writes them back to disk in an asynchronous fashion. By using such a process, dirty blocks are destaged to disk before they reach the head of the LRU chain. Hence, write misses typically do not incur the cost of a disk access to write back a dirty replaced block. Only read misses have to wait for the block to be fetched from disk. The overall I/O response time is mainly determined by the read miss access time. The *destage* process turns small random synchronous writes into large sequential asynchronous writes. In our simulations, the destage process is initiated at regular intervals. The time between two initiations of the process is the *destage period*. The write accesses issued by the destage process are scheduled progressively so that they will cause minimal interference with the read traffic.

For organizations using parity, the destage process accomplishes two purposes: it groups several dirty blocks together to perform a single multiblock I/O and frees up space in the cache by getting rid of blocks holding old data. Decreasing the destage period increases the write traffic seen by the disk. Increasing it reduces the hit ratio and increases the likelihood that the block at the head of the LRU chain is dirty which may cause a miss to wait for the replaced block to be written to disk.

One might wonder whether the destage policy used is better than the basic LRU policy in which dirty blocks are written back only when they get to the head of the LRU chain and a miss occurs. The question is even more relevant in the case of the Base and Mirror organizations in which old data blocks are not

Table 4: Default parameters.

Array size	$N = 10$
Block size	4 KB
Synchronization method	Disk First
Striping unit for RAID5	1 block
Parity placement for ParStrip	middle cylinders
For cached organizations:	
Cache size	16 MB

kept in the cache. We have compared the two policies for various cache sizes and found that the periodic destage policy always performs better for all organizations. In [12] a background process is used to write dirty blocks from the head of the LRU chain along with other dirty blocks in the cache that belong to the same track. In organizations using parity, there is a need for freeing old data blocks periodically even if the corresponding dirty block is not at the head of the LRU chain. It might be useful though to decouple the two issues by using the destage process that writes dirty blocks from the head of the LRU chain more frequently while a flushing process is only initiated from time to time to scan the entire cache and free old data blocks.

We also examine the use of parity caching in combination with a RAID4 organization. In this case, when a write is performed, the parity is computed and written to the cache instead of writing it directly to the parity disk. The parity blocks are sorted by cylinder number and spooled to the parity disk using the SCAN policy. In the case of single block accesses, what is kept in the cache is not the actual parity but the xor of the old and new data and, when the block is to be updated on the parity disk, the old parity must be read to compute the new parity. In the case of full stripe writes, the actual parity is computed and held in the cache and then written to the parity disk without reading the old parity. For partial stripe writes, either case may occur depending on the size of the request. In the case where the parity disk queue becomes large enough to occupy the entire cache, reads and writes are serviced directly from disk and writes have to wait for a block to become free in the cache in order to store the parity update.

4 Experiments

Unless otherwise specified, the parameters shown in Table 4 are used by default in the following experiments.

4.1 Synchronization

Figure 4 shows the results for the various synchronization policies for both *RAID5* and *Parity Striping*. We see that the naive strategy (SI) has significantly worse performance than the other policies and DF performs better than RF because it reduces the response time of update accesses without significantly affecting disk utilization. The variation that gives priority to the parity access achieves better performance with both the DF and RF policies. Hence, overall, DF/PR is the best synchronization strategy. For larger array sizes, the gap between the performance of the various strategies narrows because the amount of queuing is smaller for large arrays because of better load balancing in the case of RAID5 and because of the reduced correlation between increases in load in the case of Parity Striping.

4.2 Uncached Arrays

In a first experiment, we looked at non-cached organizations and measured the performance of all four organizations for different array sizes. Figure 5 shows the response time in milliseconds for values of N from 5 to 20. In the Parity Striping organization, the parity areas were placed on the middle cylinders.

For mirrored disks, the response time for writes is the largest of the response times at the two disks in the mirrored pair. Reads, however, encounter less queuing since both disks of the pair can service reads in parallel. Moreover, the shortest seek optimization[6] is used to further reduce read response time. Since there are many more reads than writes, the overall performance of mirrors is better than the Base organization. For $N = 10$, the response time of mirrors is shorter than that of the Base organization by 12% for Trace 1 and 25% for Trace 2. The reason mirrors perform better for Trace 2 in spite of the higher write fraction is their ability to split the read load over two disks which reduces queuing.

Comparing RAID5 to the basic organization, we notice that for Trace 1, there is a significant decrease in performance associated with RAID5. Given that the fraction of large requests is small, the advantage of RAID5 in terms of high transfer rates cannot be fully exploited. There are two major effects that determine the performance of RAID5: one is the the cost of small write requests and the other is the load balancing issue. To service a single-block write request, the data and parity disks have to be read to get the old data/parity; then, the new data and parity blocks have to be written to the disk. Reading the old data/parity adds an extra rotation time to the response time of the request at each of the two disks involved. However, the response time of the parity update can be affected by queuing delays at the data disk since the parity write cannot be initiated until the old data have been read. The increased cost of write requests affects also read

[6] A read is directed to the disk that has its arm nearest to request's location.

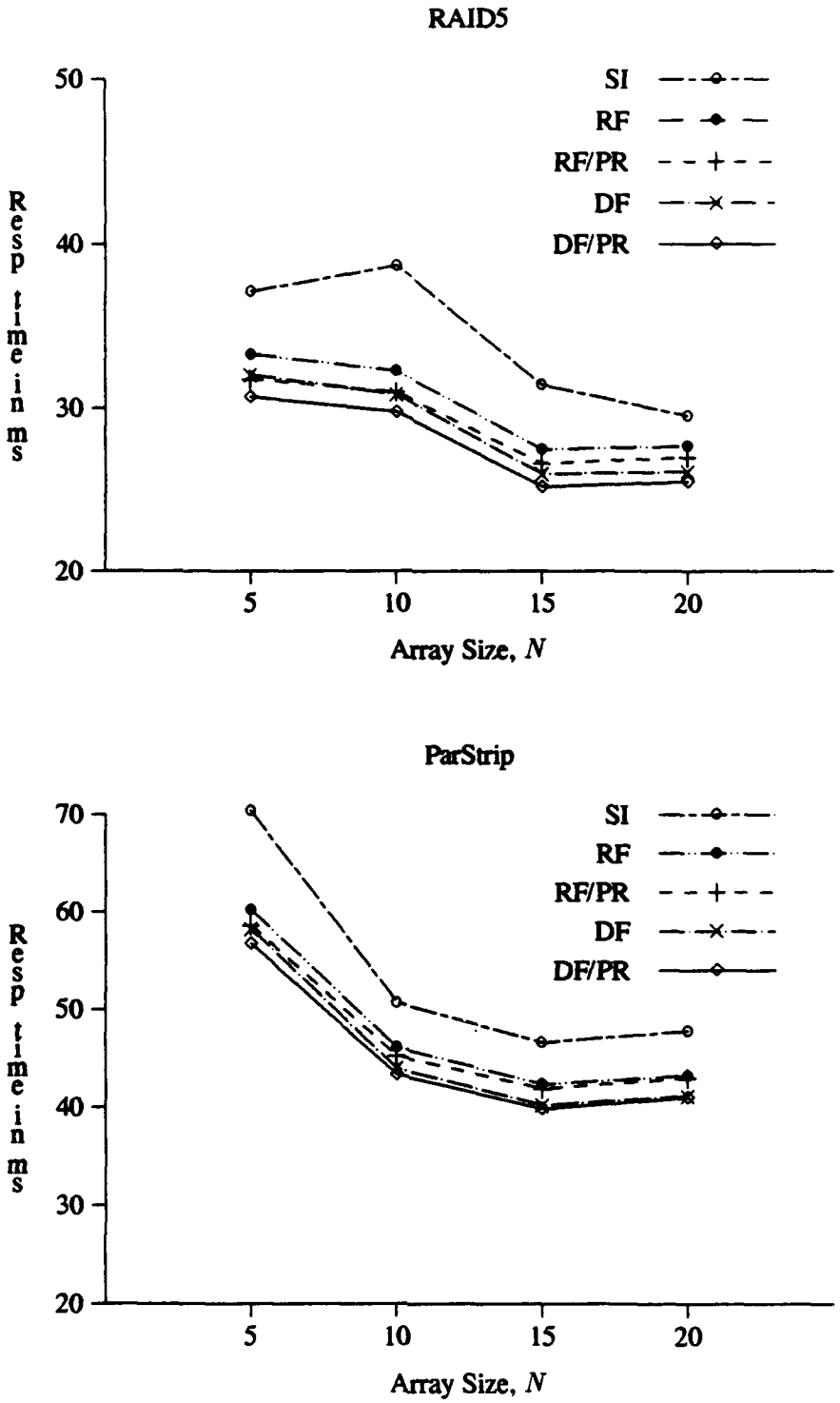
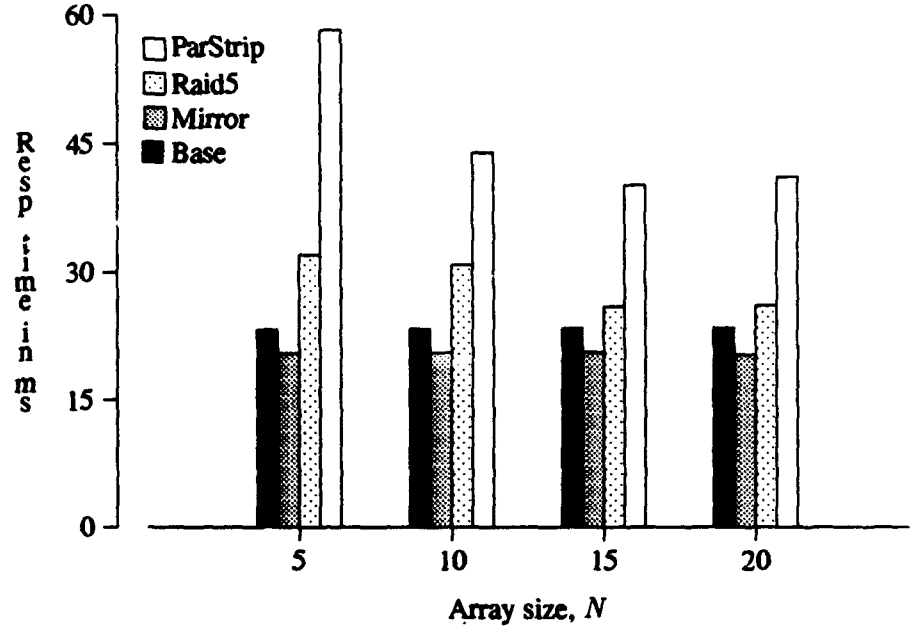


Figure 4: Response time for different synchronization methods vs. array size. (SI: Simultaneous Issue, RF: Read First, RF/PR: Read First with Priority, DF: Disk First, DF/PR: Disk First with Priority)

Trace 1



Trace 2

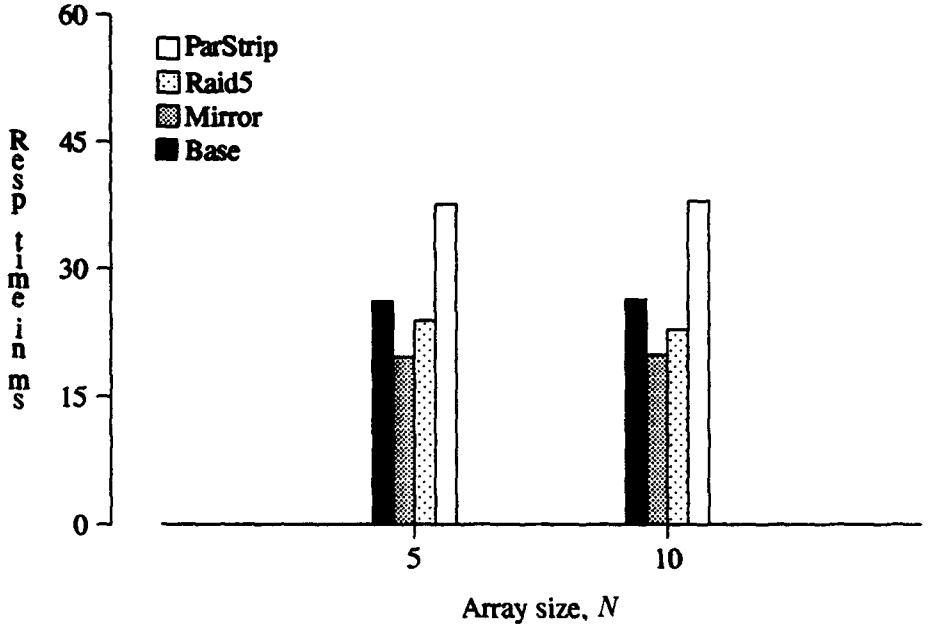


Figure 5: Response time vs. array size.

requests since it increases queuing for the disks. RAID5 balances the load over the disks in the array which reduces queuing delays. Another parameter that affects both read and write requests is *seek affinity*, which is a measure of the spatial locality that may exist among disk accesses. The higher the seek affinity, the smaller the disk arm movements. Data striping decreases seek affinity and hence increases average seek distance and seek time. In the case of Trace 1, the write penalty issue is more important than the load balancing issue. For Trace 2, load balancing seems to have a more important effect on performance than the write penalty. This is due to the fact that there is a lot of skew in the accesses to the disks in Trace 2. Note that for $N = 10$, the results for Trace 1 represent the average over 13 different arrays while for Trace 2, there is only one array. Menon and Mattson [11] found that, in the absence of disk access skew, the performance of non-cached arrays can be significantly worse than that of non-striped systems.[7] Our results confirm that the write penalty has a significant negative effect on the performance of RAID5 arrays in transaction processing systems however, in cases of high disk access skew such as in Trace 2, RAID5 may outperform non-striped systems by balancing the load and reducing queuing delays.

The difference in performance between Parity Striping and RAID5 is mainly a result of the the ability of RAID5 to balance the load over all the disks in the array. For single block accesses, the service time at the disk (seek + latency + transfer) is higher in the case of RAID5 because of decrease in seek affinity, but RAID5 more than makes up for it by reducing queuing delays. The main argument used in [7] against RAID5 is the increased disk utilization due to having many arms service a single request. This does not happen, however, if the striping unit is chosen appropriately. In transaction processing workloads, most requests are for single blocks. If the striping unit is a multiple of the block size, then most small requests are serviced by a single disk.

Note that tuning the placement of the data on disk can reduce the skew in disk accesses and, hence, reduce the gap in performance between RAID5 and Parity Striping. However, RAID5 provides a way to balance the load automatically. Figures 6 and 7 illustrate this effect. In Figure 6, the total number of accesses to each disk is plotted for Trace 1 for the Base organization, while Figure 7 plots the distribution in the case of the RAID5 organization with 4 KB striping unit. Figure 6 shows that there is a significant amount of skew in the disk access rate. Most of the skew within the array is smoothed out in the RAID5 organization.

[7] They found that the response time of non-cached RAID5 arrays was 50% higher than non-cached non-striped systems assuming 500 IO/s arrival rate with a system capacity of 100 GB over 32 data disks and a 25% write fraction.

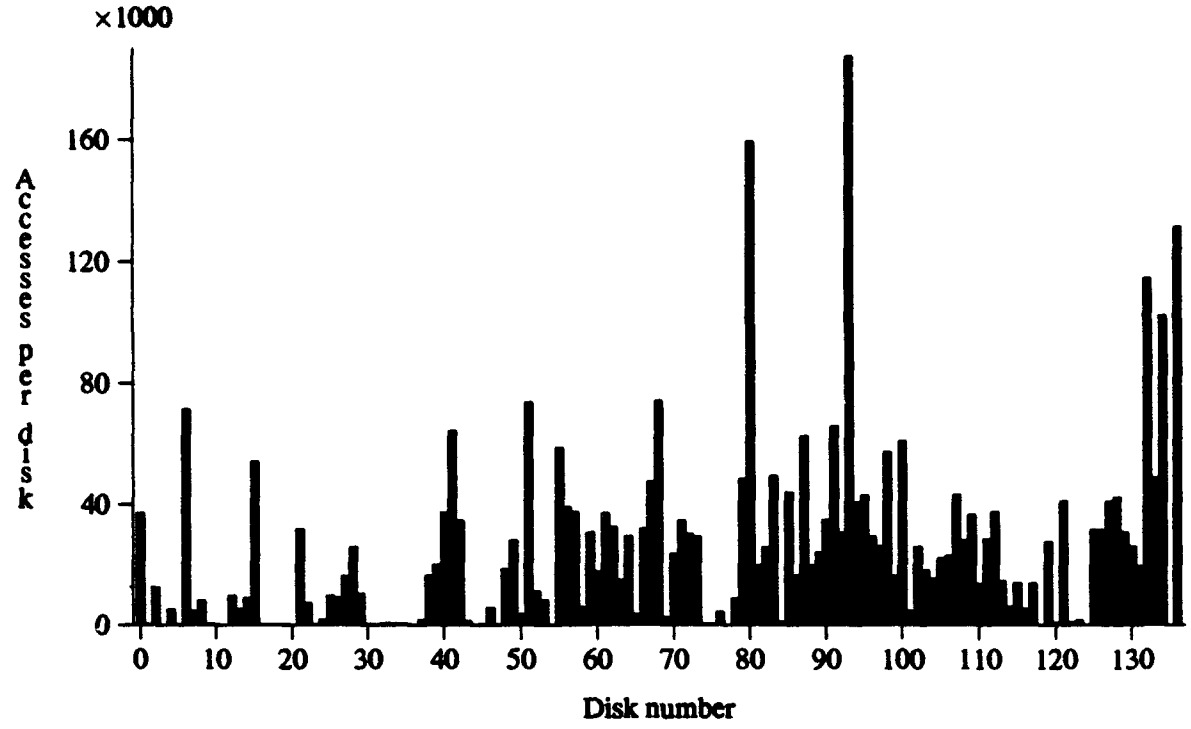


Figure 6: Distribution of accesses to disks in the Base organization (Trace 1).

4.2.1 Array Size

Changing the array size does not significantly affect the performance of the Base and Mirror organizations. There is only a very small increase in response time as the array size increases due to added channel contention since the same channel is servicing more disks. In the case of RAID5, the performance is affected by the fact that smaller arrays use more disks (for $N = 5$, there is one extra disk for every five data disks, while for $N = 10$, there is one extra disk for every ten disks). The other effect is that for large arrays the load is balanced over more disks which means that the risk of encountering large queuing delays is further reduced. For Trace 1, $N = 20$ has the lowest cost and very good performance. However, large arrays are less reliable and have worse performance during reconstruction following a disk failure.

Figure 5 shows that, for Trace 1, the Parity Striping performance deteriorates for small arrays. One cause of this behavior is the fact that the parity area becomes larger for small arrays which increases the seek distance of reads and data writes since the parity area is in the middle of the disk. This is more apparent in Trace 1 because it has a higher read fraction. The effect of the placement of the parity is analyzed in more detail in Section 4.2.3. Another cause of the performance degradation is that the Parity Striping organization aggravates the skew problem by creating a correlation between increases in load at disks in the same array. This occurs because when a hot spot appears on one disk and the disk becomes a bottleneck,

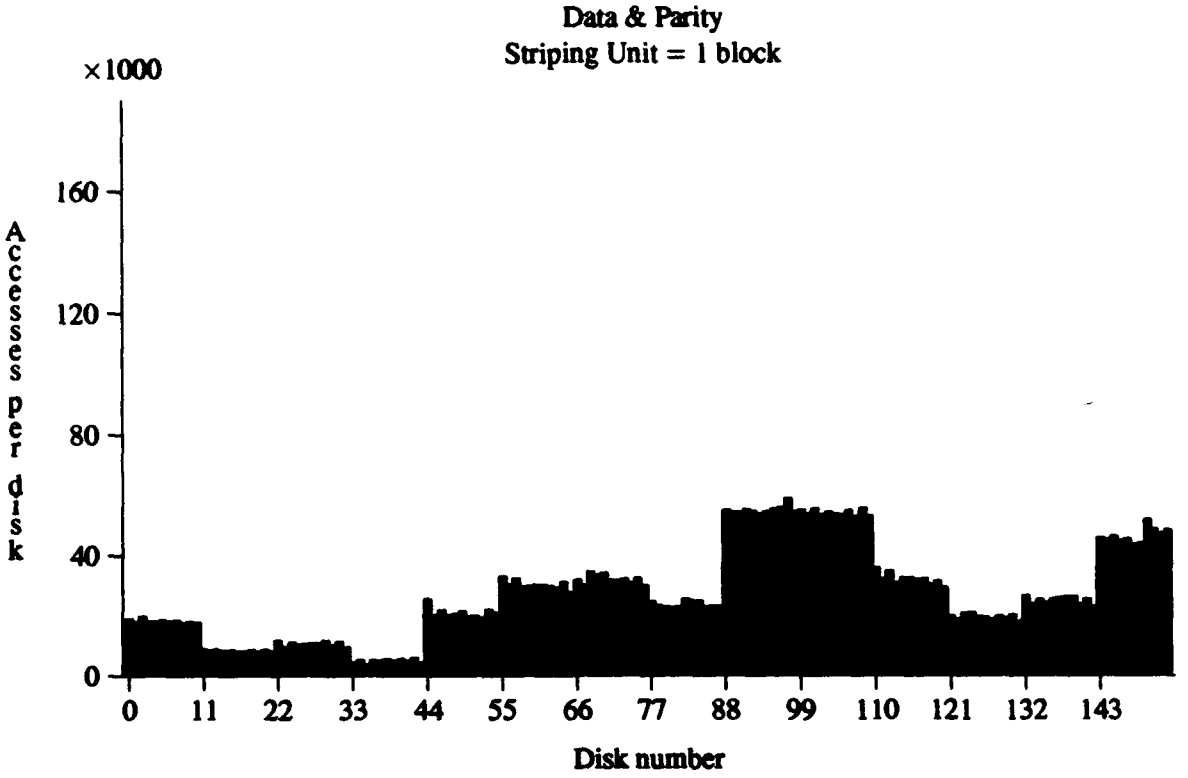


Figure 7: Distribution of accesses to disks in the RAID5 organization (Trace 1).

the disk holding the corresponding parity area also experiences increased load and possibly long queues, which, in turn, affect the performance of other disks in the array. This phenomenon is more severe for small array sizes. One possible solution to this problem would be to use a finer grain in striping the parity so that the parity update load is more balanced over the disks. Such an organization would preserve the benefits of seek affinity for read accesses and writes to the data. It also preserves other useful properties of parity striping, such as better fault containment than RAID5, control over the distribution of data over the disks and various software benefits.

4.2.2 Striping Unit in RAID5 Organizations

The striping unit for RAID5 was varied from 1 block to 64 blocks with $N = 10$. The tradeoff between small and large striping sizes is similar to the tradeoff between RAID5 and Parity Striping. Large striping sizes provide better seek affinity and reduce total disk utilization by avoiding situations in which multiple arms move to service a small multiblock request. However, they do not balance the load as well as with small striping units. Figure 8 shows the results. For Trace 1, the optimal striping unit is 8 blocks or 32 kBytes. There is little difference in performance, however, between values from 1 to 16 blocks. For Trace 2, the

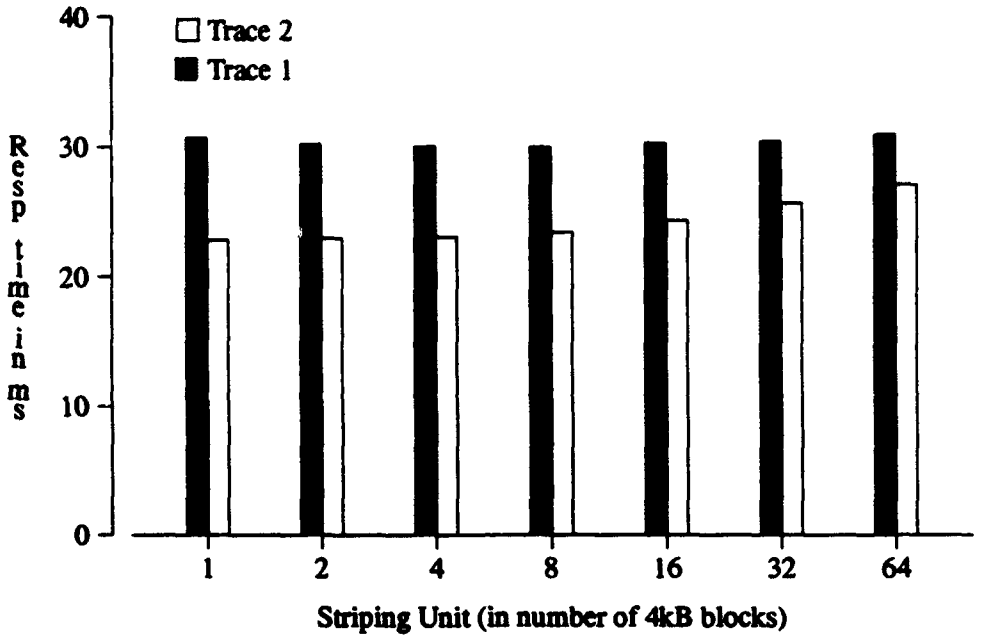


Figure 8: Response time vs. striping unit for RAID5.

optimal striping unit is 1 block which indicates that there is more need for load balancing in the case of Trace 2. For a striping size of 32 blocks or more, the performance starts degrading significantly and, for very large striping units, it approaches that of Parity Striping.

4.2.3 Parity Placement in Parity Striping Organizations

In [7], it was suggested that since the parity area is accessed frequently, it should be placed on the cylinders at the center of the disk. We found that this does not always improve performance especially for small arrays where the parity areas are quite big and when the workload has a high read-write ratio. A simple model can be used to explain this effect. Assuming that accesses are uniformly distributed over all disks in the array and that accesses to a given disk are uniformly distributed over all data areas on the disk, the access rate to any one of the N data areas on each disk is equal to $1/N^2$ times the total access rate to the array while the access rate to any given parity area is w/N times the total access rate to the array, where w is the fraction of accesses that are writes. Hence the parity areas are accessed more often than the data areas if and only if $w > 1/N$. In the workload of Trace 1, we have $w = 0.1$. Hence according to the model, for $N > 10$, the parity area should be placed in the middle of the disk while for $N < 10$, it should be placed at the end of the disk. In Figure 9 the results for the two placements are shown for various values of N. For Trace 1, we observe that the rule established by the above model is verified except that the cutoff point

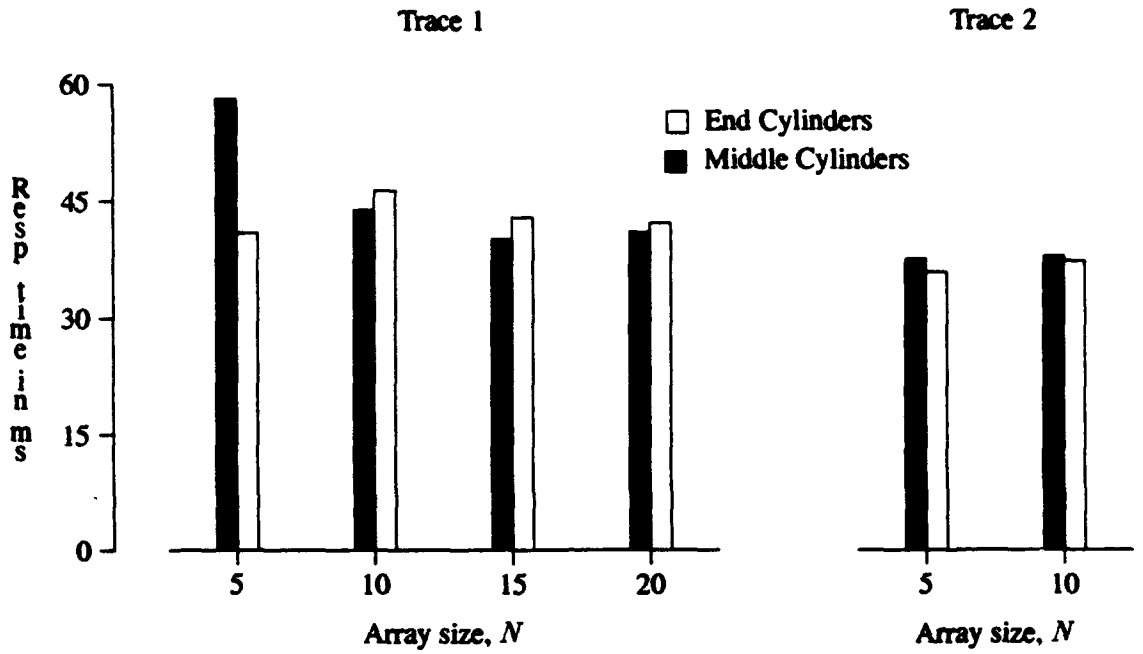


Figure 9: Comparison of parity placements for Parity Striping.

occurs somewhere between $N = 5$ and $N = 10$ (probably closer to 10 than to 5 given the large difference in performance seen for $N = 5$). For Trace 2, the rule does not seem to be satisfied which means that the uniform access assumptions break down in this case. However, we see that the middle cylinder placement is worse for small N which confirms the trend suggested by the above model.

4.2.4 Modifying Trace Speed

In order to study the performance of the various organizations at higher or lighter loads, we have conducted an experiment in which the trace was speeded up by a factor of 2 and another one where the trace was slowed down by a factor of two. Note that the workloads obtained by speeding up the trace do not reflect the characteristics of any real system. Doubling the processor speed does not imply that I/O's will be issued twice as fast since transactions may have to wait for one I/O to finish before issuing another one. RAID5 response time degrades gracefully as the load increases. RAID5 does even better than mirrors when the load doubles. The response times for Parity Striping and to a lesser degree that of the Base organization degrade severely as trace speed is doubled. Figure 10 shows the results. For Trace 2 and for a trace speed of 0.5, the base organization performs better than RAID5 because at low trace speed there is little queuing and RAID5 looses its load balancing advantage.

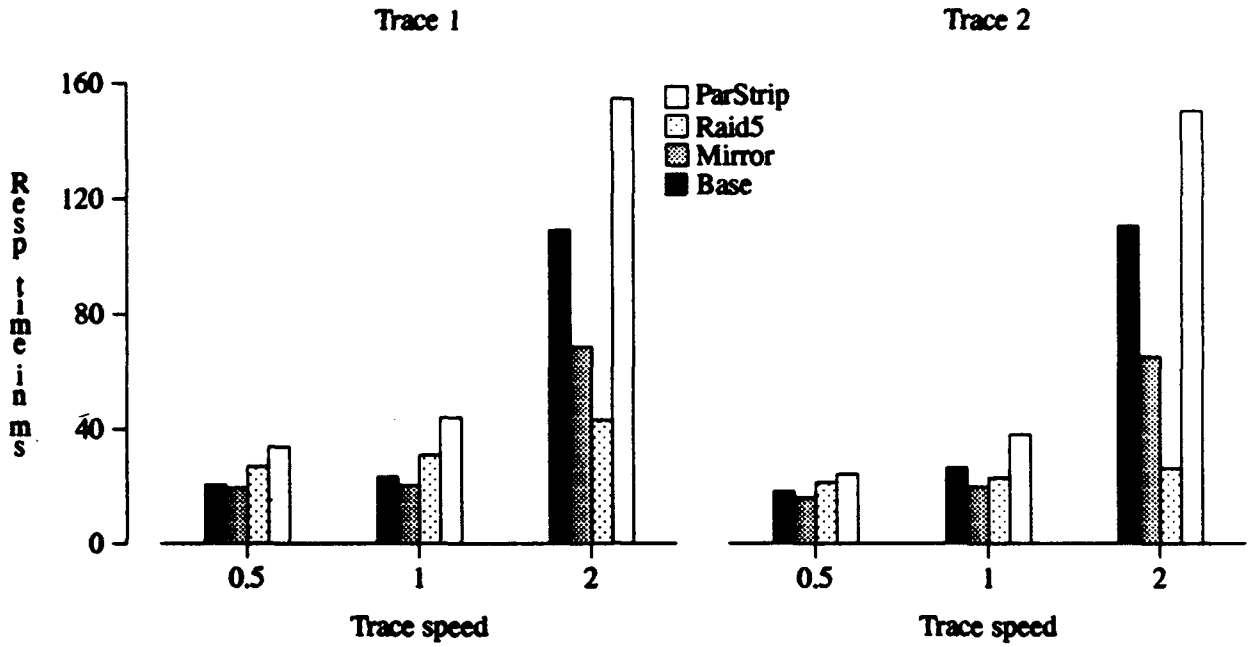


Figure 10: Response time vs. trace speed.

4.3 Performance of Cached Organizations

The cache hit ratio is slightly lower for RAID5 and Parity Striping because of the space held by the old blocks in the cache. The read and write hit ratios are plotted in Figure 11 for both traces and both for organizations using parity (RAID5, Parity Striping) and for those not using parity (Base, Mirror). Multiblock accesses are counted as hits only if all of the blocks requested are in the cache. For Trace 1, the write hit ratio is almost one for large caches because blocks are usually read by the transaction before being updated. For Trace 2, the write hit ratio starts at about 20% and increases to over 60%. The workload in Trace 2 seems to contain large working sets that require a relatively large I/O cache. The read hit ratio is relatively low for a small cache size ($\approx$ 9% for Trace 1 and $<$ 1% for Trace 2 for an 8 MB cache) but it increases to about 54% for Trace 1 and 40% for Trace 2 for a cache size of 256 MB. The effect on hit ratio of keeping the old blocks in the cache is minimal. The difference between the parity and the non-parity organizations is always less than 1% for writes. For reads, the difference in hit ratio is higher. For Trace 1, the hit ratio of the parity organizations is 6% lower for an 8 MB cache but the difference goes down to 2% for a 16 MB cache and keeps decreasing for higher cache sizes. For Trace 2, the relative difference is highest for the 32 MB case, where the hit ratio goes from 4.6% for Base to 3.5% for RAID5, but the gap narrows significantly for higher cache sizes.

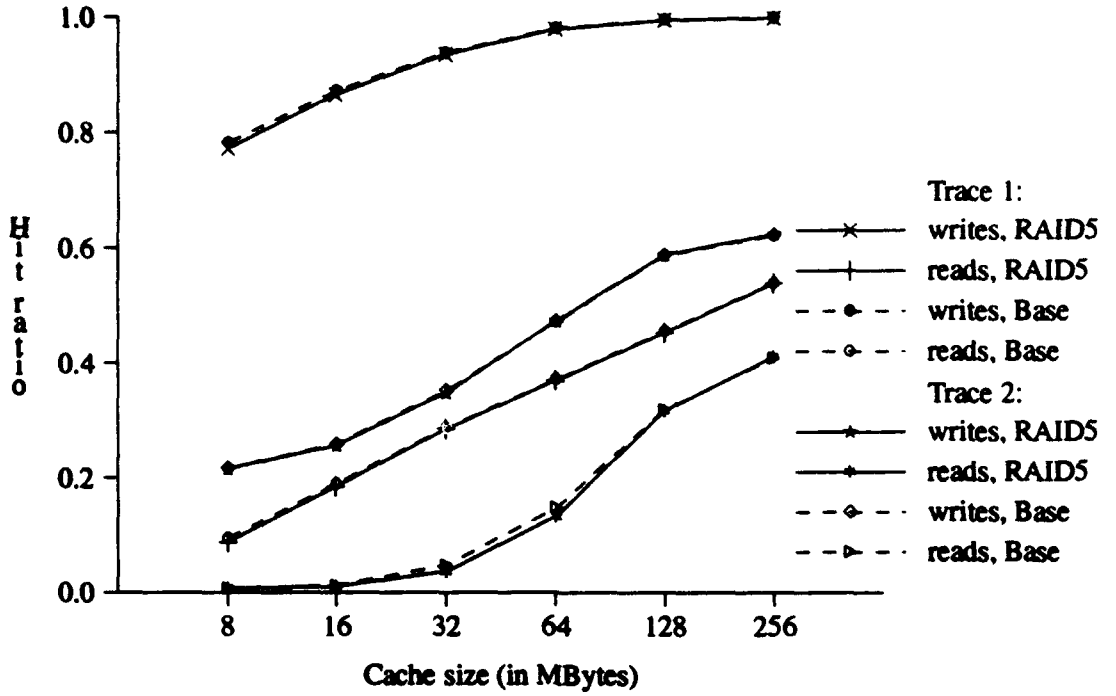
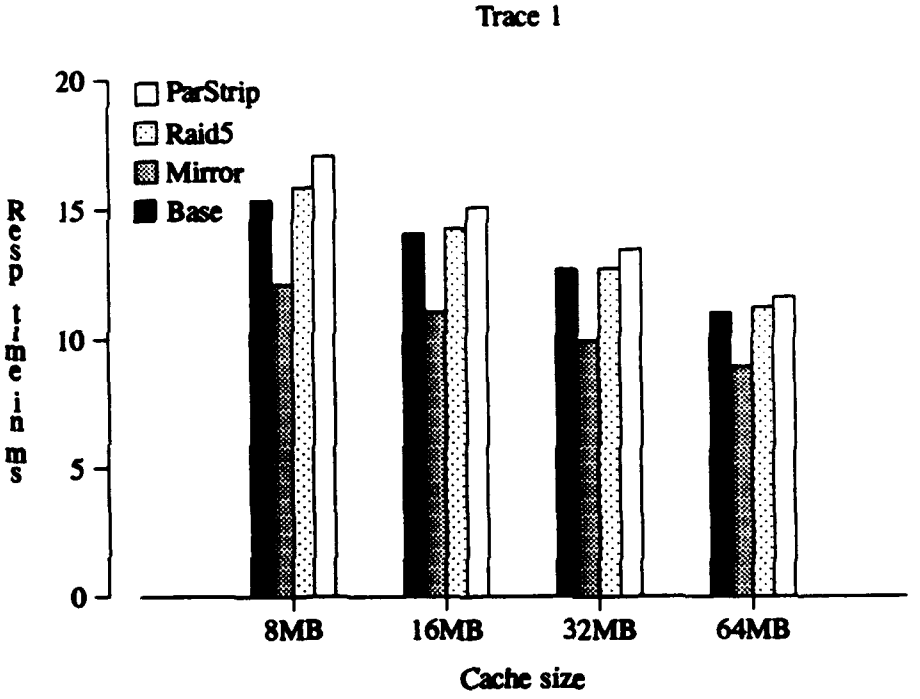


Figure 11: Hit ratio vs. cache size.

4.3.1 Cache Size

The response time results are shown in Figure 12. All organizations benefit from larger cache sizes. The performance of mirrors is still better than the Base organization. Since each of the disks in the mirrored pair sees the same destage traffic as the corresponding disk in the base organization, the contribution of the destage traffic to read miss access time is the same in both organizations. In addition mirrors service reads faster because the read load is shared between the two disks in the mirrored pair and because of the shorter seek optimization. For a 16 MB cache size, mirrors perform 22% better than the Base organization for both traces.

The gap in performance between RAID5 and the Base organization reduces considerably in the case of Trace 1 because the larger cost of writes in RAID5 does not affect the overall response time directly but only through its contribution to read miss waiting time. Write costs are also reduced by the fact that old blocks are kept in the cache and that the number of actual writes decreases because multiple updates to the same block while it is in the cache result in one actual write to disk. As cache size increases, the gap gets even smaller in relative terms because the difference in miss ratios becomes smaller, the likelihood of having the old block in the cache when the write is destaged becomes higher and the probability (which should be already very small) of having to wait for a replaced page to be written to the disk on a cache miss becomes even smaller thus further reducing the contribution of the higher costs of write in RAID5 to the response time. For a



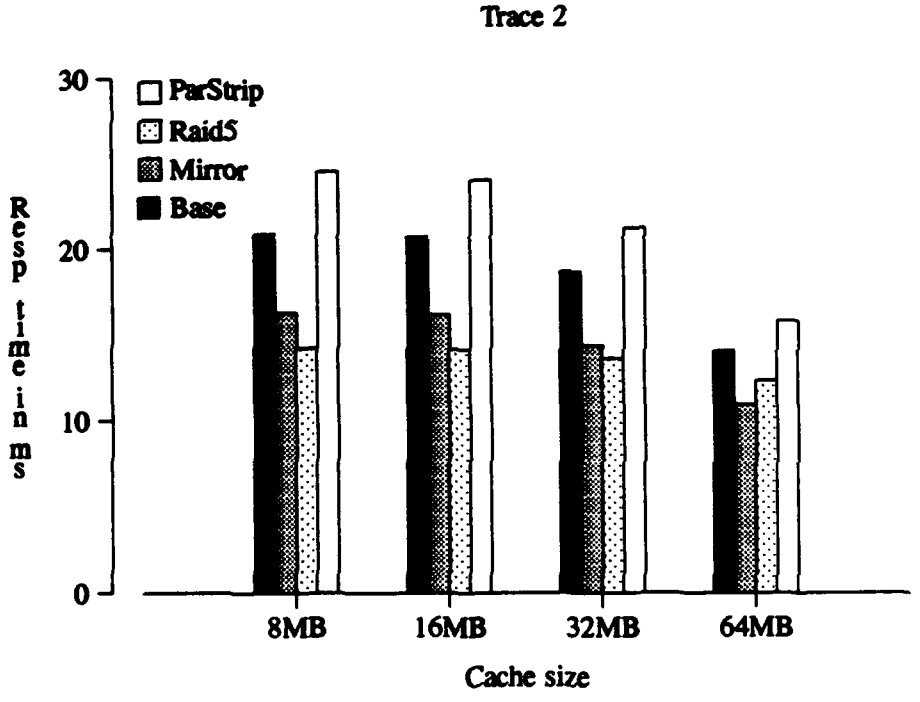


Figure 12: Response time vs. cache size.

cache size of 16 MB, RAID5's performance is only about 1% worse than that of the Base organization. In the case of Trace 2, RAID5 does even better than in the non-cached case especially for small cache sizes since the write penalty is practically eliminated while the need for load balancing remains because of the low hit ratio. RAID5 even surpasses mirrored disks for cache sizes less than 64 MBytes. The gap between RAID5 and Base narrows as cache size increases because the need for load balancing decreases.

These results are in agreement with Menon and Mattson's analysis [11] which showed that, in the absence of disk access skew, the performance of RAID5 arrays with cached controllers is only a few percentage points (3 to 9%) worse than the Base organization. This is to be compared with as much as 50% degradation in the case of non-cached RAID5 arrays according to their analysis.

RAID5 still does better than Parity Striping. The gap between the two narrows for Trace 1 mainly because of the reduced load at the disks which makes RAID5's load balancing advantage less important in determining response time. For Trace 2, the difference is still significant because of the low hit ratio.

4.3.2 Array Size

In Figure 13, we compare three organizations with different array sizes but with the same total cache size. For $N = 5$, the cache size in each array is 8 MB while for $N = 10$, the cache size is 16 MB and for $N = 15$, the cache size is 24 MB. For the Base and Mirror organizations, the performance improves slightly in the case of Trace 1 in the larger array in spite of the higher channel contention. This implies that a large shared cache for 10 disks is better than two partitioned caches for every five disks. For Trace 2, the effect of channel contention is more important than the increase of hit ratio due to the shared cache. In the case of RAID5 and Parity Striping, the number of disk arms and the load balancing issue have more effect on performance than the difference in hit ratio between a global and a partitioned cache or the channel contention.

4.3.3 Striping Unit

The response time of the cached RAID5 organization is plotted in Figure 14 as a function of the striping unit. For Trace 1, the optimal striping unit in this case is 16 blocks or 64 kBytes compared with 8 blocks for the non-cached organization. The reason for the difference is that the load on the array is lighter in the cached organization; therefore, the need for load balancing is not as high. This makes larger striping units more efficient because they can take better advantage of seek affinity and reduce disk utilization on multiblock accesses. For Trace 2, the optimal striping unit is still 1 block as in the non-cached organization. This is the case because of the low hit ratio for this trace.

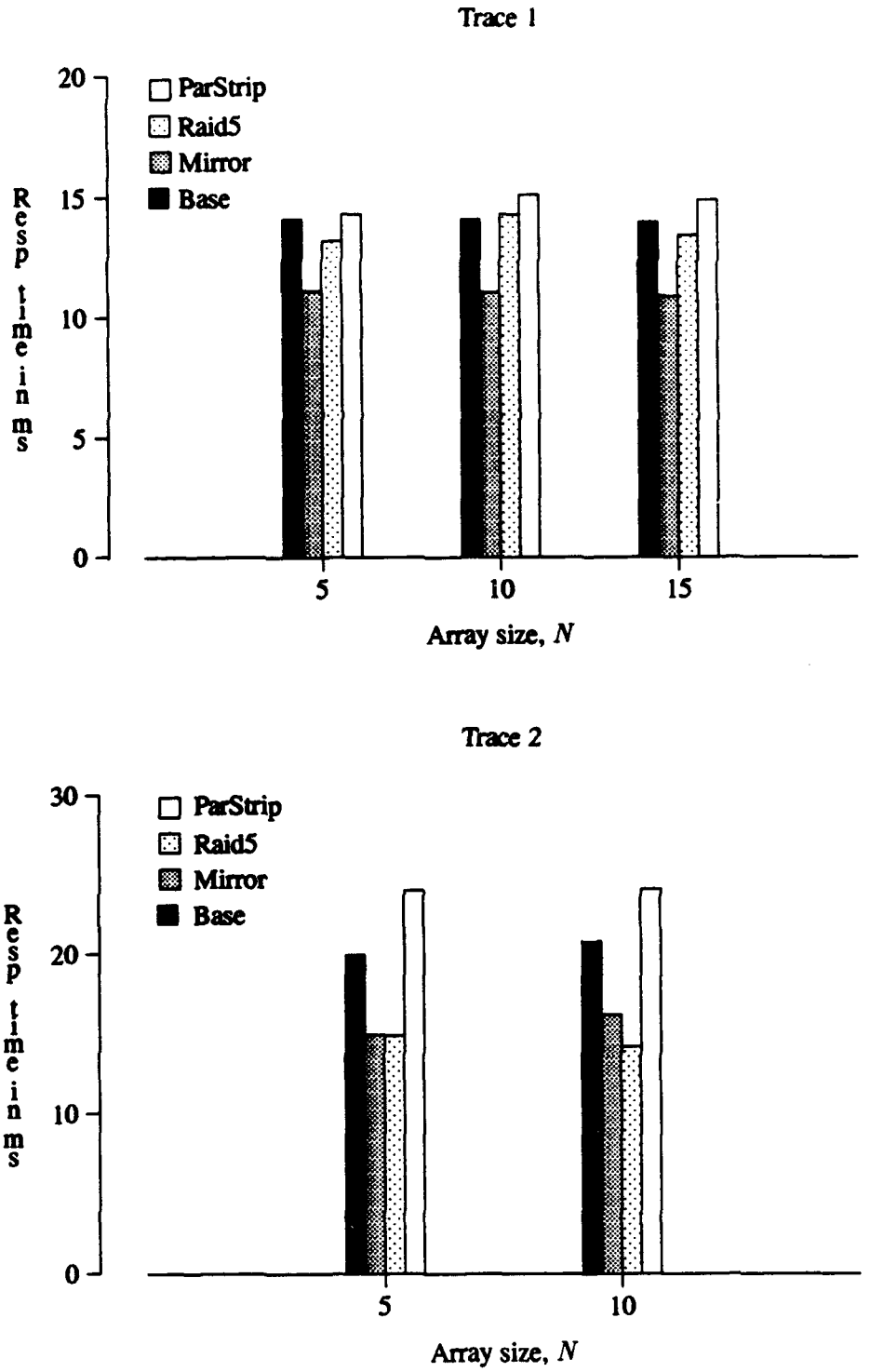


Figure 13: Response time vs. array size for cached organizations.

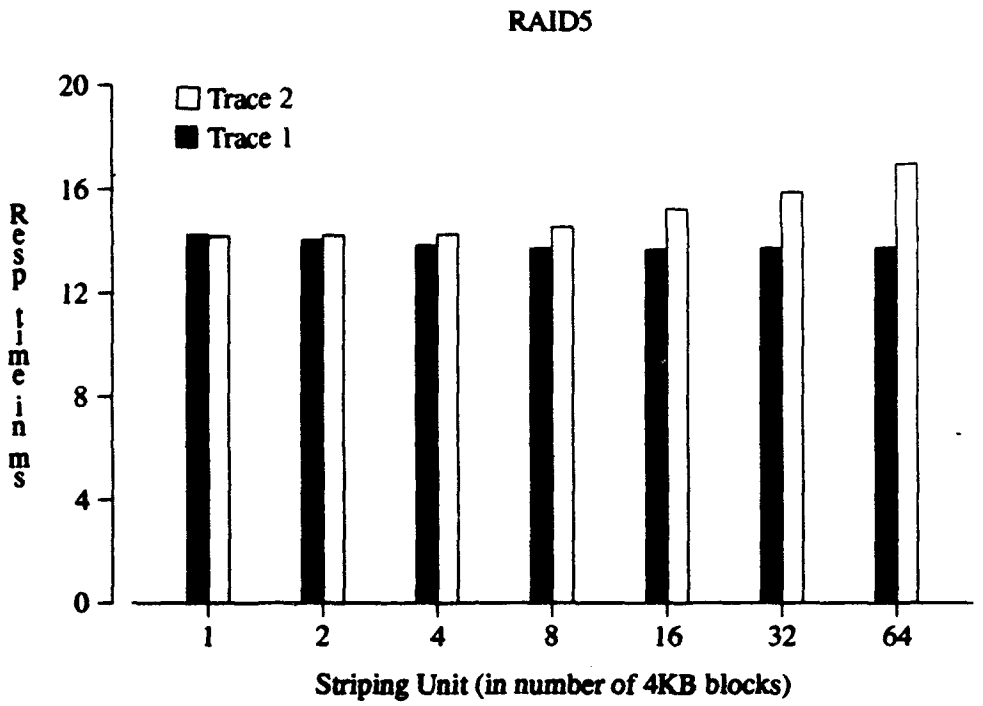


Figure 14: Response time vs. striping unit for RAID5 with cache.

4.4 Parity Caching

In this section, we examine a RAID4 organization in which the parity resides on a separate disk in the array. The parity updates are buffered in the controller cache before being written to the dedicated parity disk. We compare the performance of such an organization to the RAID5 organization in which only data blocks are cached. We look at the effect of various parameters such as cache size, array size, trace speed, and striping unit. One benefit of using the RAID4 with parity caching organization is the fact that read miss accesses, will not have to wait behind parity accesses which include an extra rotation due to the need to read the old block before updating it. Another benefit is the reduced average seek distance because parity blocks are not placed in between data blocks. Disadvantages include the reduced hit ratio in the cache due to the space occupied by parity blocks and the fact that the number of disk arms available for servicing the synchronous read miss accesses decreases by one. Another issue is the fact that the parity disk may become a bottleneck for the entire array if its queue grows and fills up the cache. A necessary condition for the parity disk to keep up with the update traffic is $N\lambda w < \mu$, where λ is the arrival rate at each of the data disks, w is the write fraction and μ is the service rate at the parity disk.

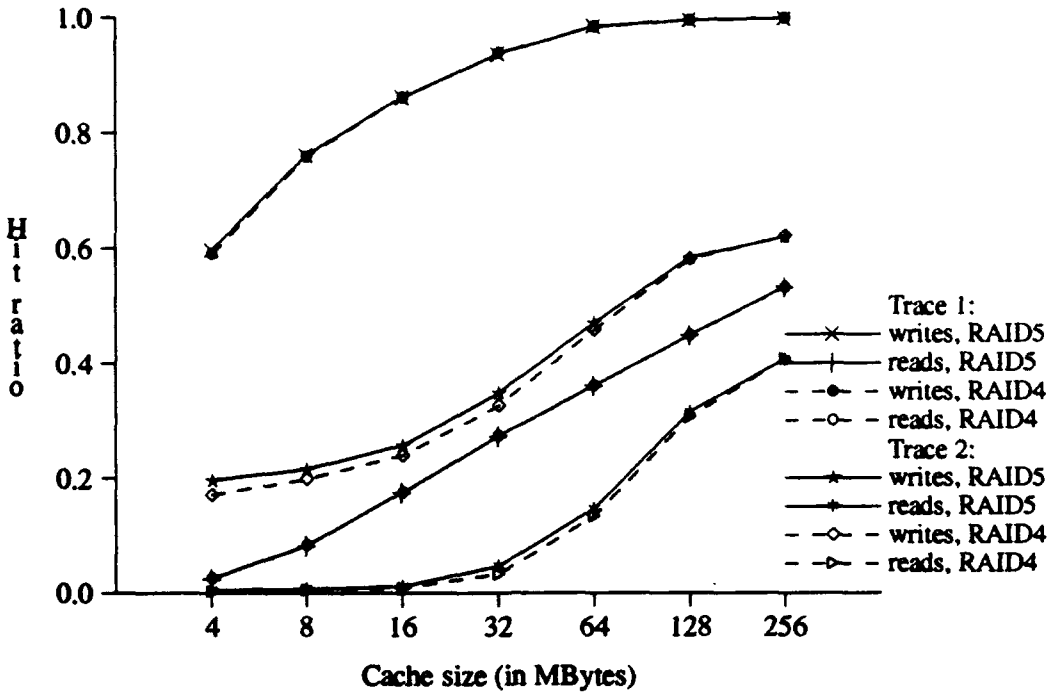


Figure 15: Hit ratio with parity caching vs. cache size.

4.4.1 Cache Size

The read and write hit ratios are plotted in Figure 15 for RAID5 and for RAID4 with parity caching. The effect on hit ratio of buffering the parity blocks in the cache in the RAID4 organization is minimal for Trace 1. For Trace 2, the gap is wider; however, the relative difference is significant only in the region where the hit ratio is quite small and therefore has little effect on overall performance.

The response time results are shown in Figure 16. For Trace 1, the difference between the two organizations is small but the RAID4 organization always does better. The hit ratio is actually lower for RAID4 than for RAID5 but the fact that parity updates are performed on a separate disk and do not interfere with the read accesses seems to outweigh the effect of the lower hit ratio. As cache size increases, the gap between the two organizations becomes even smaller in relative terms because the probability of a synchronous I/O due to a read miss becoming smaller. The response time of RAID4 is 2% lower for a cache size of 8 MBytes and about 1% lower for a cache size of 16 MBytes.

In the case of Trace 2, RAID4 performs much better than RAID5, especially at small cache sizes. This is due to the higher percentage of writes in Trace 2 ($\approx$ 30%) compared to Trace 1 ($\approx$ 10%) and to the lower hit ratio in the case of Trace 2. The use of RAID4 and parity caching significantly reduces the cost of writes and their effect on read miss response time. For a 16 MByte cache the response time for RAID4 is 15% shorter than for RAID5. The gap between the two organizations narrows significantly as the cache size

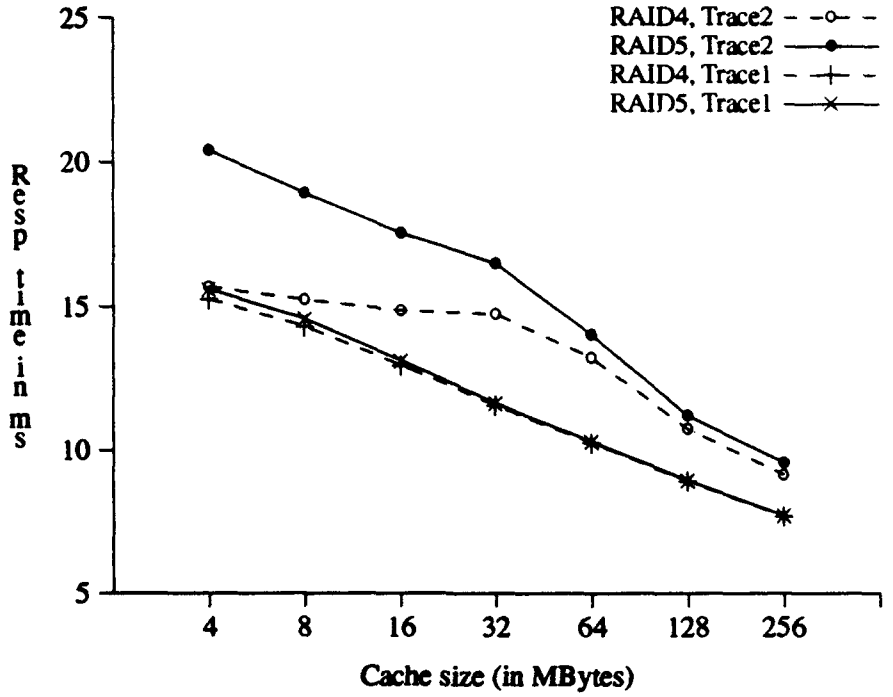


Figure 16: Response time vs. cache size.

increases.

4.4.2 Array Size

In Figure 17, we compare the two organizations for three different array sizes while maintaining the same total cache size. For $N = 5$, the cache size in each array is 8 MB while for $N = 10$, the cache size is 16 MB and for $N = 20$, the cache size is 32 MB. For $N = 5$, RAID5 performs better than RAID4 for both traces because, with RAID4, fewer disks are available to service read requests (5 out 6 disks service read requests for $N = 5$ compared with 10 out of 11 for $N = 10$). This implies that dedicating one disk per array for parity updates does not pay off for small arrays. For Trace 1, we see that, going from $N = 10$ to $N = 20$, the gap between RAID4 and RAID5 widens. This is due to the fact that in RAID4, for larger N, a larger proportion of the disks can service read accesses (the ratio $N/N + 1$ increases with N). The load on the parity disk increases as N increases but this does not seem to significantly affect the performance of RAID4.

4.4.3 Trace Speed

The gap between the two organizations widens as the load increases. Figure 18 shows the results. In the

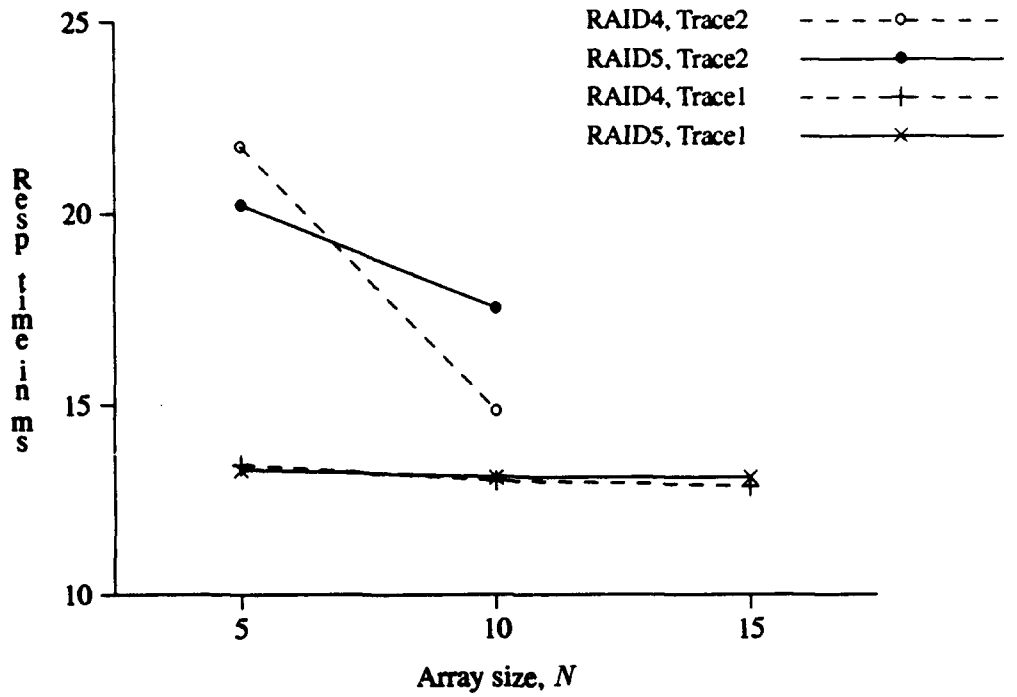


Figure 17: Response time vs. array size.

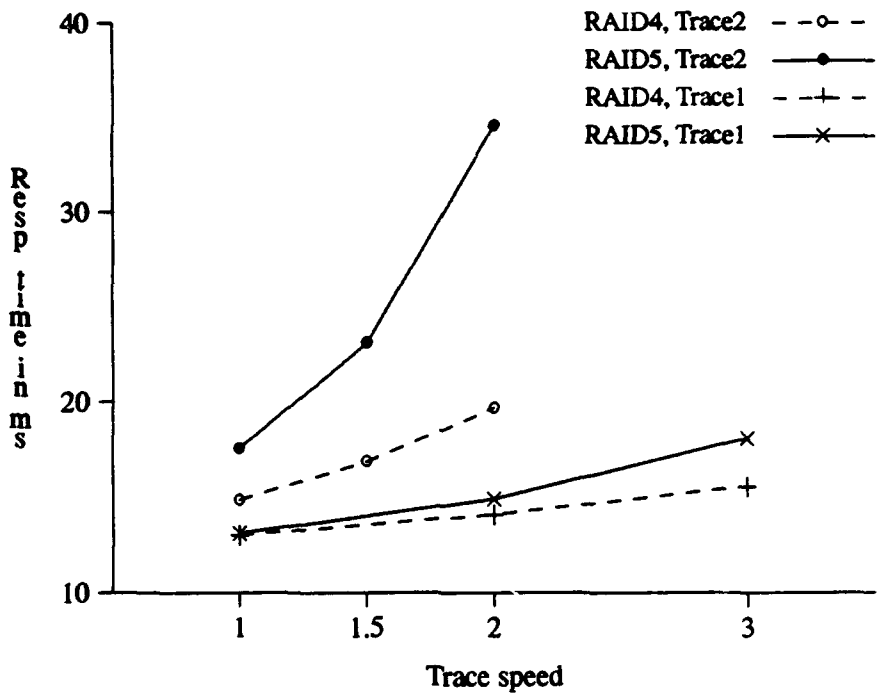


Figure 18: Response time vs. trace speed.

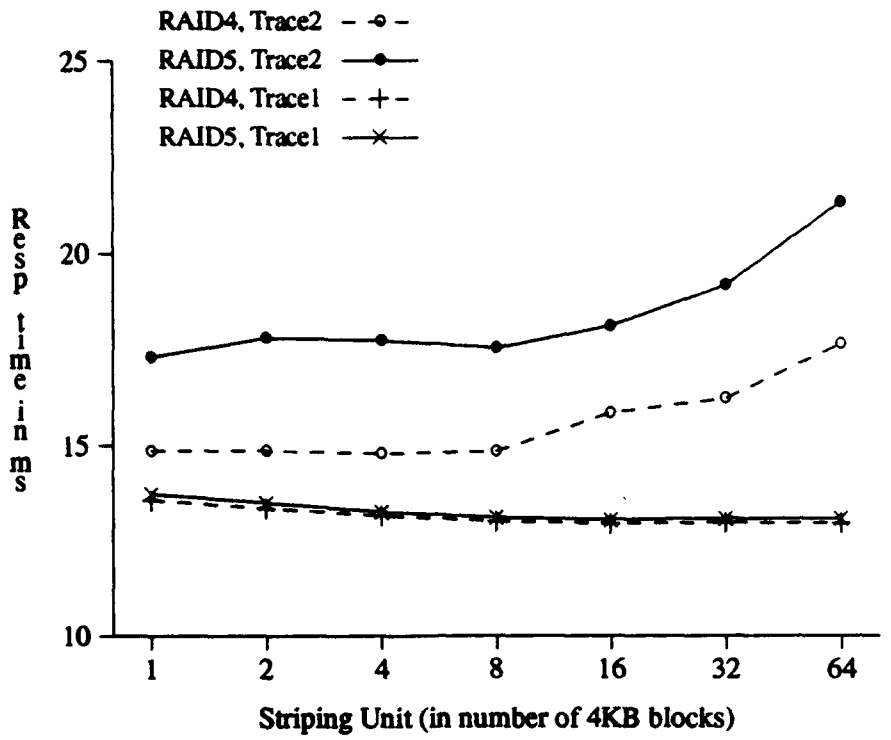


Figure 19: Response time vs. striping unit.

case of Trace 2, RAID5's performance degrades significantly at high loads. The increasing load on the parity disk in the RAID4 organization did not create a bottleneck. There are periods in the traces where the parity disk queue becomes large enough to fill most of the cache in which case writes to the disk have to wait for an empty slot to open in the cache for writing the parity. However, these heavy load periods are rare and do not last very long; and there are sufficient idle periods in the traces for the parity disk to catch up and empty its queue which is stored in the cache.

4.4.4 Striping Unit

When disk utilization is high such as in the case of Trace 2, load balancing becomes an important issue, hence, a smaller striping unit is preferred. The response time of the two organizations is plotted in Figure 19 as a function of the striping unit. The shapes of the curves for Trace 1 and Trace 2 (RAID4 case) are predictable. The response time decreases at first as the striping unit increases because seek affinity is better exploited to reduce average seek time. But as the striping unit becomes larger, the load becomes more unbalanced which causes long delays at some disks and increases the average response time. The optimal striping unit is lower for Trace 2 than for Trace 1 because of the higher disk utilization for Trace 2. The shape of the first part (striping unit $\leq$ 4) of the curve for Trace 2 (RAID5 case) is not predictable and is

probably due to the particular reference stream and block layout encountered in that trace which causes some heavily accessed blocks to land on the same disk(s) for the striping units 2 and 4.

5 Conclusions

We used traces from commercial transaction processing systems to evaluate the performance of two redundant disk array organizations and compare them to mirrored disks and to non-redundant non-arrayed systems (Base organization). The I/O workload is dominated by single block I/O's and contains a significant amount of skew in the distribution of accesses to disks. We evaluated both cached and non-cached organizations. We found that RAID5 outperforms Parity Striping in all cases because of its load balancing capabilities.

In non-cached organizations, RAID5 and Parity Striping may perform significantly worse than the Mirror and Base organizations because of the high cost of individual writes. For an organization with 10 data disks per array, RAID5's response time is 32% worse than the Base organization for one of the traces. It was also found that, because of the large amount of skew in disk accesses found in the workload, large RAID5 arrays perform better than smaller arrays by balancing the load more evenly over the disks. By speeding up the trace, it was shown than RAID5 behaves better than the other organizations under very high loads. For Parity Striping organizations, we found that placement of the parity area on the disk can affect performance significantly and we derived a simple rule for placing the parity in a way that minimizes seek times.

For cached organizations, we found that all organizations benefit from higher cache sizes. The write hit ratio is much higher than the read hit ratio and in one of the benchmarks it is close to 1 for cache sizes over 32 MBytes. The read hit ratio on the other hand keeps increasing steadily as cache size increases. RAID5's performance is as good or better than the Base organization's performance in cached organizations. A 16MB cache practically eliminates the RAID5 write penalty. For one of the traces, RAID5 performance goes from 32% worse than the Base organization in the non-cached case to only about 1% worse in the cached cache.

In our evaluation of cached organizations, we also studied a RAID4 organization that uses the controller cache to buffer parity updates before sending them to the parity disk. We found that RAID4 with parity caching generally performed better than RAID5 for array sizes of 10 or more. The improvement achieved is a function of the percentage of writes in the I/O workload and the load (arrival rate) at the disks. The load at the disks is a function of the cache size and the amount of locality in the workload. The improvement in response time varied from 1% for the first benchmark with a cache size of 16 MBytes per (10+1)-disk array to as much as 15% for the other benchmark with the same cache size. At smaller caches sizes, the number of I/O's going to the disks increases and so does the benefit of parity caching. We have studied the effect of

array size on the performance of RAID4 with parity caching. We found that at higher array sizes, RAID4's performance improves because a larger proportion of the disks can service read requests while the parity disk is able to keep up with the increased load. For a small array size ($N = 5$), RAID5 performs better than RAID4 with parity caching because it uses more disk arms to service reads. We have also experimented with speeding up the trace and found that parity caching can effectively remove the bottleneck on the parity disk in RAID4 even at high loads. For both benchmarks used, it was found that although the parity disk queue can grow at times to occupy most of the cache, this did not occur often and there were sufficient idle periods for the parity disk to catch up.

There are two important issues that we have touched upon that warrant further investigation. One of these issues is the optimal destage policy when old data blocks are kept in the cache. Another issue is the use of a smaller striping unit for the parity in order to balance the parity update load in the Parity Striping organization.

Acknowledgements

The authors would like to thank Kent Treiber, Ted Messinger, Honesty Young and Robert Morris of IBM Almaden Research Center for providing the traces and for answering numerous questions.

References

[1] M. Y. Kim, "Synchronized disk interleaving," *IEEE Transactions on Computers*, vol. C-35, pp. 978–988, Nov. 1986.

[2] M. Livny, S. Khoshafian, and H. Boral, "Multi-disk management algorithms," in *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 69–77, May 1987.

[3] K. Salem and H. Garcia-Molina, "Disk striping," in *Proceedings of the IEEE International Conference on Data Engineering*, pp. 336–342, Feb. 1986.

[4] A. Reddy and P. Banerjee, "An evaluation of multiple-disk I/O systems," *IEEE Transactions on Computers*, vol. 38, pp. 1680–1690, Dec. 1989.

[5] D. Patterson, G. Gibson, and R. Katz, "A case for redundant arrays of inexpensive disks (RAID)," in *Proceedings of the ACM SIGMOD Conference*, pp. 109–116, June 1988.

[6] D. Bitton and J. Gray, "Disk shadowing," in *Proceedings of the 14th International Conference on Very Large Data Bases*, pp. 331–338, Sept. 1988.

[7] J. Gray, B. Horst, and M. Walker, "Parity striping of disk arrays: Low-cost reliable storage with acceptable throughput," in *Proceedings of the 16th International Conference on Very Large Data Bases*, pp. 148–161, Aug. 1990.

[8] P. M. Chen, G. A. Gibson, R. H. Katz, and D. A. Patterson, "An evaluation of redundant arrays of disks using an Amdahl 5890," in *Proceedings of the ACM Sigmetrics Conference on Measurement and Modeling of Computer Systems*, pp. 74–85, May 1990.

[9] S. Chen and D. Towsley, "The design and evaluation of RAID 5 and parity striping disk array architectures," *Journal of Parallel and Distributed Computing*, pp. 58–74, Jan. 1993.

[10] E. K. Lee and R. H. Katz, "The performance of parity placements in disk arrays," *IEEE Transactions on Computers*, vol. 42, pp. 651–664, June 1993.

[11] J. M. Menon and R. L. Mattson, "Performance of disk arrays in transaction processing environments," in *Proceedings of the 12th International Conference on Distributed Computing Systems*, June 1992.

[12] A. L. N. Reddy, "A study of I/O system organizations," in *Proceedings of the International Symposium on Computer Architecture*, pp. 308–317, 1992.

[13] G. R. Ganger, B. L. Worthington, R. Y. Hou, and Y. N. Patt, "Disk subsystem load balancing: Disk striping vs. conventional data placement," in *Proceedings of the 1993 Hawaii International Conference on System Sciences*, pp. 40–49, Jan. 1993.

[14] A. Bhide and D. Dias, "RAID architectures for OLTP." IBM T.J. Watson Research Center Technical Report, 1992.

[15] J. O'Brien, "RAID 7 architecture features asynchronous data transfers," *Computer Technology Review*, Winter 1991.

[16] D. Stodolsky, G. Gibson, and M. Holland, "Parity logging: Overcoming the small write problem in redundant disk arrays," in *Proceedings of the 20th International Symposium on Computer Architecture*, pp. 64–75, May 1993.

[17] P. M. Chen and D. A. Patterson, "Maximizing performance in a striped disk array," in *Proceedings of the International Symposium on Computer Architecture*, pp. 322–331, May 1990.

[18] A. L. N. Reddy, "Reads and writes: When I/O's aren't quite the same," in *Proceedings of the Hawaii International Conference on System Science*, pp. 84–92, 1992.